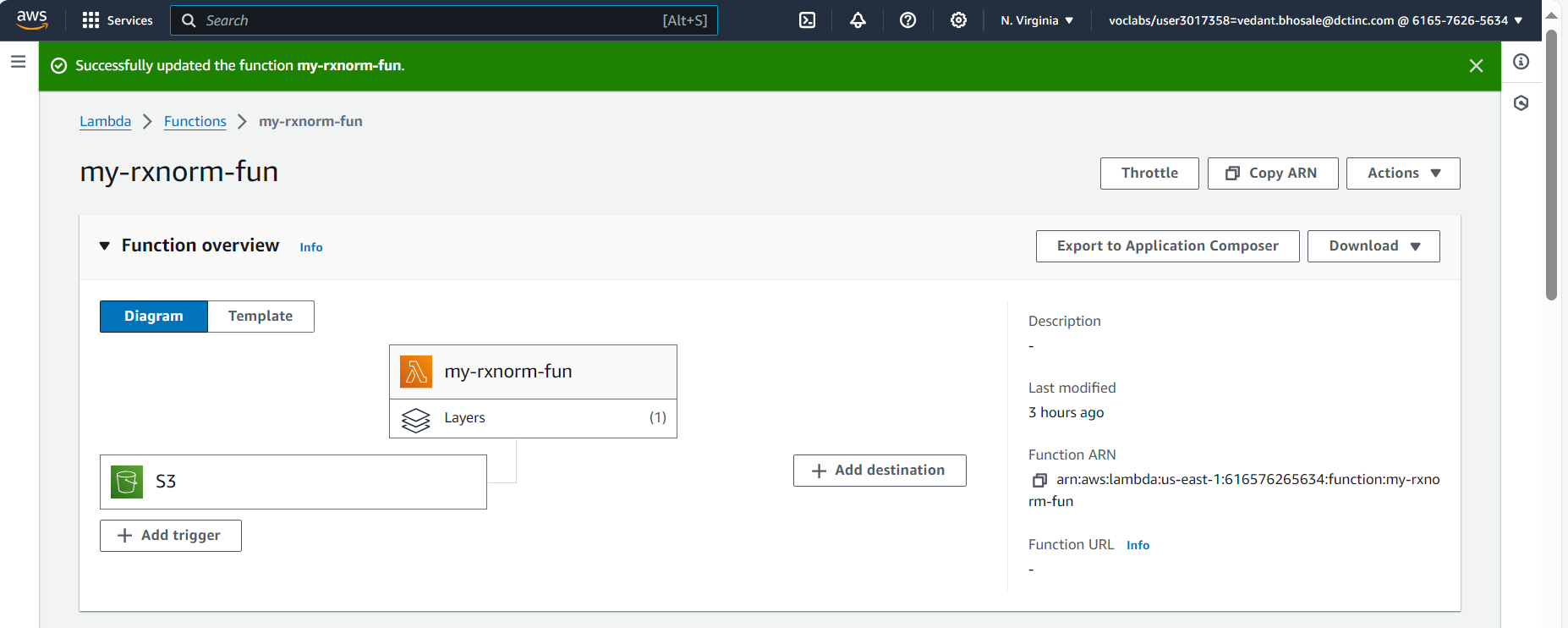
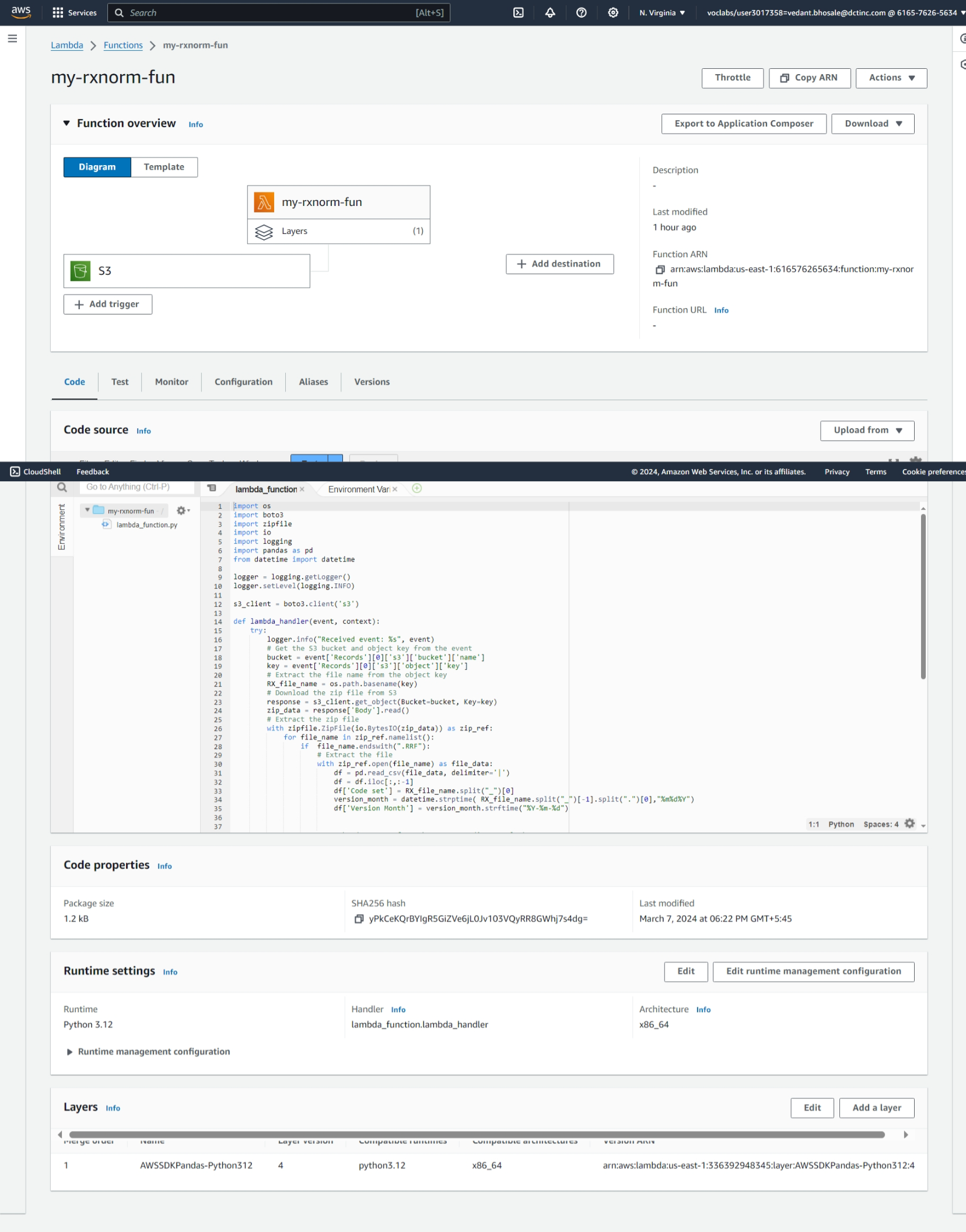
Created lambda function which triggers when we put zip files in source folder



A screenshot of a computer

Description automatically generated

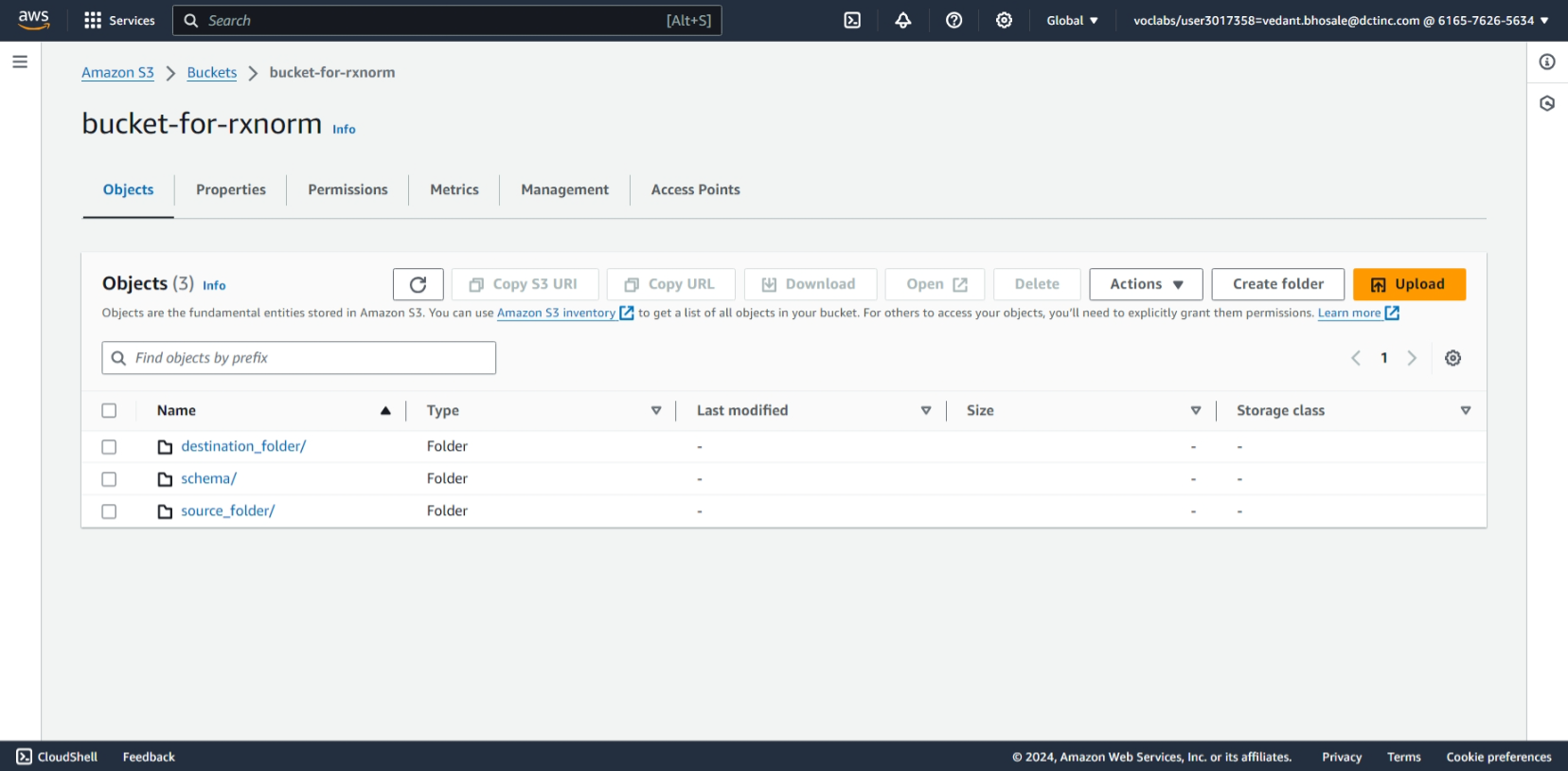
Created a pandas layer to create dataframes from files and assign them headers



Bucket where zip file is uploaded in source\_folder

Header excel file in schema

And converted files are stored in destination\_folder



Lambda code to unzip the zipped files then assigning them headers from headers excel file and saving them in destination folder.

import os

import boto3

import zipfile

import io

import logging

import pandas as pd

from datetime import datetime

logger = logging.getLogger()

logger.setLevel(logging.INFO)

s3\_client = boto3.client('s3')

def lambda\_handler(event, context):

    try:

        logger.info("Received event: %s", event)

        # Get the S3 bucket and object key from the event

        bucket = event['Records'][0]['s3']['bucket']['name']

        key = event['Records'][0]['s3']['object']['key']

        # Extract the file name from the object key

        RX\_file\_name = os.path.basename(key)

        # Download the zip file from S3

        response = s3\_client.get\_object(Bucket=bucket, Key=key)

        zip\_data = response['Body'].read()

        # Extract the zip file

        with zipfile.ZipFile(io.BytesIO(zip\_data)) as zip\_ref:

            for file\_name in zip\_ref.namelist():

                if  file\_name.endswith(".RRF"):

                    # Extract the file

                    with zip\_ref.open(file\_name) as file\_data:

                        df = pd.read\_csv(file\_data, delimiter='|')

                        df = df.iloc[:,:-1]

                        df['Code set'] = RX\_file\_name.split("\_")[0]

                        version\_month = datetime.strptime( RX\_file\_name.split("\_")[-1].split(".")[0],"%m%d%Y")

                        df['Version Month'] = version\_month.strftime("%Y-%m-%d")

                    # Get header names from the corresponding Excel sheet

                    sheet\_name = os.path.splitext(os.path.basename(file\_name))[0]  # Assuming file names match sheet names

                    header\_file\_key = 'schema/RxNorm\_Header.xlsx'               # Change to your actual file name

                    header\_df = read\_excel\_from\_s3(bucket, header\_file\_key, sheet\_name)

                    if header\_df is not None:

                        df.columns = header\_df.iloc[:, 0].values  # Assuming headers are in column A

                    # Convert DataFrame to CSV format

                    csv\_data = df.to\_csv(index=False)

                    # Upload the CSV file to another folder in the same S3 bucket

                    destination\_key = f"destination\_folder/{os.path.basename(file\_name.replace(".RRF",".csv"))}"

                    s3\_client.put\_object(Bucket=bucket, Key=destination\_key, Body=csv\_data)

        return {

            'statusCode': 200,

            'body': 'Extraction and file upload completed'

        }

    except Exception as e:

        logger.error("Error: %s", e, exc\_info=True)

        return {

            'statusCode': 500,

            'body': f'Error: {e}'

        }

def read\_excel\_from\_s3(bucket, key, sheet\_name):

    try:

        response = s3\_client.get\_object(Bucket=bucket, Key=key)

        excel\_bytes = response['Body'].read()

        df = pd.read\_excel(io.BytesIO(excel\_bytes), sheet\_name=sheet\_name, header=None)

        return df

    except Exception as e:

        logger.error("Error reading Excel file: %s", e)

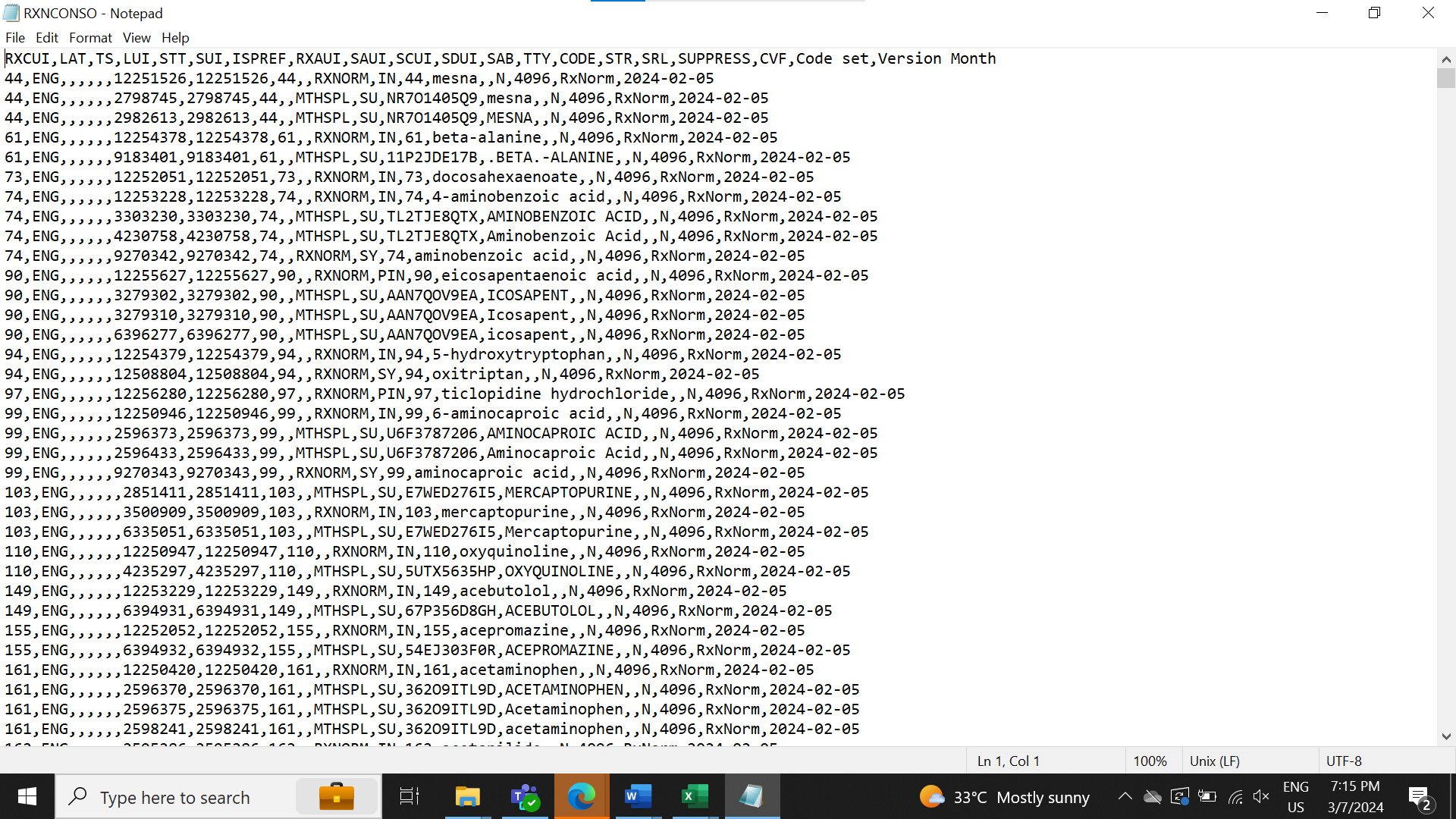
        return None

Modified files in destination folder

A screenshot of a computer

Description automatically generated

Files with headers assigned



A screenshot of a computer

Description automatically generated